Confinement de fautes Byzantines dans les réseaux multi-sauts asynchrones

Alexandre Maurer, Sébastien Tixeuil

Confinement de fautes Byzantines dans les réseaux multi-sauts asynchrones

Alexandre Maurer[1] et Sébastien Tixeuil[1]

[1] *UPMC Sorbonne Universités (France), {alexandre.maurer, sebastien.tixeuil}@lip6.fr*

On considère le problème de la diffusion d'information dans un réseau multi-saut asynchrone, en présence de fautes Byzantines : certains nœuds du réseau peuvent adopter un comportement arbitraire. Les protocoles de diffusion existant reposent sur une forte connectivité du réseau. Dans cet article, nous proposons un protocole adapté aux réseaux faiblement connecté. Nous donnons une méthodologie pour construire un ensemble de nœuds *fiable*, et évaluons ses performances sur une topologie de grille, avec des fautes Byzantines aléatoirement distribuées.

Keywords: Tolérance aux fautes Byzantines, protocole de diffusion, réseaux multi-sauts, réseaux asynchrones

1 Introduction

Motivations On considère le problème de la diffusion dans un réseau : un nœud peut vouloir diffuser une information particulière à l'ensemble du réseau. Lorsque la taille du réseau augmente, il devient très probable d'avoir des nœuds *fautifs*, qui ne respectent pas le protocole de diffusion. Nous considérons ici le modèle de faute le plus général possible : le modèle Byzantin. Un nœud Byzantin peut envoyer des messages totalement arbitraires à ses voisins ; par conséquent, le pire comportement possible doit être envisagé. Ce modèle a été introduit par [LSP82], puis largement étudié dans les réseaux totalement connectés. Nous nous intéressons ici aux réseaux multi-sauts, où l'information doit être relayée de voisin en voisin. Nous laissons de côté les approches cryptographiques [DFS05], qui reposent sur une infrastructure fiable et ne sont donc pas totalement distribuées. Le protocole [Koo04] repose sur un système de vote local, et fonctionne correctement si chaque nœud a moins d'une fraction $1/4\pi$ de voisins Byzantins. Le protocole [NT09] utilise un vote sur plusieurs chemins nœud-disjoints, et permet de tolérer k Byzantins si le réseau est $(2k+1)$-connecté.

Notre approche Les approches existantes considèrent le pire placement possible d'un nombre donné de Byzantins. De plus, leur critère est que tous les nœuds corrects doivent communiquer de façon fiable. Par conséquent, pour tolérer davantage de Byzantins, il faut augmenter la connectivité du réseau, ce qui est une contrainte lourde dans un monde où les réseau deviennent de plus en plus grand. Notre idée est de proposer un protocole de diffusion adapté à des réseaux de grande taille, mais faiblement connecté. Pour y parvenir, nous affaiblissons les exigences usuelles. D'une part, nous retirons à l'adversaire son pouvoir de localisation : on ne considère plus le placement au pire cas des Byzantins, mais un placement aléatoire. Cette modélisation est réaliste dans certains cas, comme celui d'une surcouche pair-à-pair : les nœuds qui joignent le réseau (y compris les Byzantins) ne choisissent pas leur localisation. D'autre par, nous cherchons à obtenir une communication fiable entre une *majorité* de nœuds corrects. Le critère sera donc la probabilité, pour deux nœuds corrects choisis au hasard, de communiquer fiablement.

Organisation de l'article Nous proposons un nouveau protocole de diffusion basé sur des *zones de contrôle*, puis nous donnons une méthodologie pour déterminer un ensemble *fiable* de nœuds corrects, pour une distribution donnée de nœuds Byzantins. Nous utilisons ensuite cette méthode pour évaluer le critère précédent, pour une distribution uniforme de Byzantins sur un réseau à topologie de grille, où chaque nœud possède au plus 4 voisins.

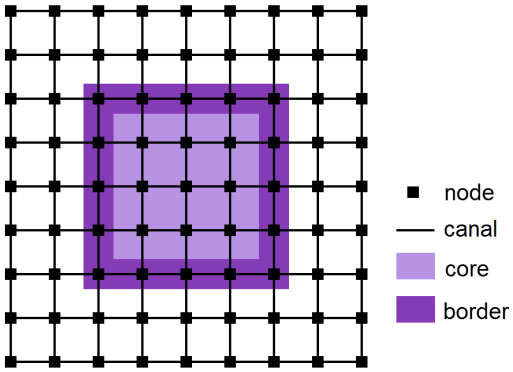

FIGURE 1: Exemple de zone de contrôle

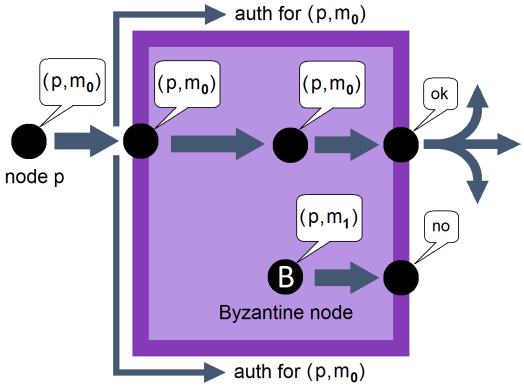

FIGURE 2: Principe d'une zone de contrôle

2 Protocole de diffusion

Dans cette partie, nous proposons un protocole de diffusion qui limite statistiquement l'impact des fautes Byzantines. Après avoir décrit le protocole, nous donnons deux propriétés qui permettent de déterminer un ensemble *fiable* de nœuds du réseau. Nous utilisons ensuite ces propriétés pour faire l'évaluation expérimentale d'un réseau à topologie de grille (faiblement connectée).

2.1 Description informelle

Soit un réseau décrit par un graphe (G,E). G est l'ensemble des *nœuds* du réseau, et E l'ensemble des *canaux* : $\forall\{p,q\} \in E$, les nœuds p et q peuvent échanger directement des messages (on dira alors qu'ils sont *voisins*). Le réseau est *asynchrone* : tout message envoyé finira par être reçu, mais cela peut prendre un temps indéterminé. On fait l'hypothèse que, lorsqu'un nœud p reçoit un message d'un voisin q, il sait qu'il a été envoyé par q.

On s'intéresse au problème de la diffusion : un nœud quelconque p souhaite diffuser un message m à l'ensemble du réseau. Dans l'idéal, p envoie (p,m) à ses voisins, qui envoient à leur tour (p,m) à leurs voisins, et ainsi de suite, jusqu'à ce que tous les nœuds aient reçu (p,m). Alors, le réseau entier sait que p a diffusé m. Cependant, certains nœuds peuvent être *Byzantins* et adopter un comportement arbitraire. Par exemple, un nœud Byzantin peut diffuser (p,m'), avec $m' \neq m$, pour faire croire au réseau que p a diffusé m'. Remarquons qu'un seul nœud Byzantin peut ainsi mentir sur chaque nœud p, et donc déstabiliser l'intégralité du réseau.

Nous proposons donc un protocole de diffusion qui limite l'action des nœuds Byzantins. Ce protocole est basé sur des *zones de contrôle*, qui agissent comme des filtres pour les messages Byzantins. Une zone de contrôle est constituée de deux ensembles de nœuds, le *bord* et le *coeur*, de telle sorte que le bord isole le coeur du reste du réseau (voir Figure 1). L'idée est la suivante : lorsqu'un message veut pénétrer dans le coeur, une *autorisation* est diffusée sur le bord. Par la suite, lorsque le message voudra sortir du coeur, cette autorisation lui sera demandée. Ainsi, si un nœud Byzantin situé dans le coeur diffuse un faux message (p,m'), alors que p n'est pas dans le coeur, ce message n'obtiendra jamais l'autorisation pour sortir du coeur (à condition, bien entendu, que le bord soit constitué de nœuds corrects). Cette idée est illustrée dans la Figure 2. L'idée sous-jacente est de combiner un grand nombre de zones de contrôle, s'interpénétrant les unes les autres, afin de limiter au maximum l'action des nœuds Byzantins.

2.2 Description formelle

Soit Ctr un ensemble de zones de controle, une zone z étant décrite par les ensembles de nœuds $core(z)$ et $border(z)$ (son coeur et son bord). Ces ensembles sont tels que, si l'on supprime les nœuds de $border(z)$, $core(z)$ est totalement isolé du reste du réseau. Chaque nœud p possède un message $p.m_0$ à diffuser, et un ensemble $p.myCtr$ de zones de controles telles que $\forall z \in p.myCtr$, $p \in border(z)$. Il possède également trois ensembles dynamiques $p.Wait$ (messages en attentes d'autorisation), $p.Auth$ (autorisations reçues) et $p.Acc$

(messages acceptés). Ces ensembles sont initialement vides. On distinguera les messages *standards* de la forme (p,m) (voir plus haut) et les messages d'*autorisation* (p,m,z) (autorisation du message (p,m) pour la zone de contrôle z).

Chaque nœud p correct exécute l'algorithme suivant (les attributs m_0, $myCtr$, $Wait$, $Auth$ et Acc correspondent ici à ceux de p) :

1. Initiallement, p envoie (p,m_0) à ses voisins, ajoute (p,m_0) à Acc, et $\forall z \in myCtr$, p envoie (p,m_0,z) à ses voisins.
2. Lorsque p reçoit un message standard (s,m) d'un voisin q : si $(s,m) \in Acc$, il l'ignore ; sinon, il ajoute (s,m,q) à $Wait$.
3. Lorsque p reçoit un message d'autorisation (s,m,z) d'un voisin q : si $(s,m,z) \in Auth$ ou $q \notin border(z)$, il l'ignore ; sinon, il ajoute (s,m,z) à $Auth$ et envoie (s,m,z) à ses voisins.
4. Lorsqu'un triplet $(s,m,q) \in Wait$ vérifie la condition suivante : $\forall z \in myCtr$ tel que $q \in core(z)$ et $s \notin core(z)$, on a $(s,m,z) \in Auth$; p ajoute (s,m) à Acc, envoie (s,m) à ses voisins, et $\forall z \in myCtr$, p envoie (s,m,z) à ses voisins.

2.3 Propriétés du protocole

Soit un réseau sur lequel est implémenté le protocole précédent. Supposons qu'un observateur omniscient connaisse la position des nœuds Byzantins. On souhaite alors déterminer un ensemble *fiable* de nœuds, qui parviennent toujours à communiquer entre eux et ne sont jamais trompés par les Byzantins. C'est l'objet des deux théorèmes suivants.

Soit $Corr$ l'ensemble des nœuds corrects (non-Byzantin). On dit qu'un nœud p *accepte* (s,m) lorsque (s,m) est ajouté à $p.Acc$. Un message (s,m) est *correct* si $m = s.m_0$; sinon il est *faux*. Un chemin *correct* est une série de nœuds $(p_1,\ldots,p_n)$ tel que p_k et p_{k+1} sont voisins. Un ensemble S de nœuds est *communicant* si $\forall (p,q) \in S^2$, q finit toujours par accepter $(p,p.m_0)$.

Théorème 1 *Soit un ensemble Z de zones de contrôle. Soit* $Cores = \cup_{z\in Z} core(z)$ *et* $Borders = \cup_{z\in Z} border(z)$. *Si tous les nœuds Byzantins sont dans Cores, et que Cores et Borders sont disjoints, alors tout nœud v tel que* $v \notin Cores$ *n'acceptera jamais de faux message.*

Théorème 2 *Soit S un ensemble communicant, et v un nœud correct ayant un voisin* $u \in S$. *Soit Z l'ensemble des zones de contrôle z telles que* $u \in core(z)$ *et* $v \in border(z)$. *Si,* $\forall z \in Z$, *il existe un chemin correct contenu dans* $border(z)$, *reliant v et un nœud* $w \in S$ *: alors* $S \cup \{v\}$ *est également communicant.*

Les preuves de ces théorèmes sont détaillées dans [MT11]. Le Théorème 1 permet de déterminer un ensemble S_1 de nœuds qui ne peuvent accepter que des messages corrects. Le Théorème 2 permet de construire, nœud par nœud, un ensemble communicant. Pour initialiser cette construction, il suffit de remarquer que tout nœud correct v forme un ensemble communicant $\{v\}$. Alors, $S_3 = S_1 \cap S_2$ est un ensemble *fiable* au sens défini plus haut.

3 Évalutation expérimentale

On considère un réseau avec n_B nœuds Byzantins aléatoirement distribués, selon une loi uniforme. On souhaite évaluer la probabilité $P(n_B)$ que deux nœuds du réseau, choisis au hasard, communiquent de façon fiable. Pour cela, on utilise une méthode Monte-Carlo : on génère un grand nombre de distributions aléatoires de n_B nœuds Byzantins, et pour chacune d'elles, on détermine un ensemble *fiable* S_3 en utilisant les Théorèmes 1 et 2. Puis on choisit deux nœuds au hasard : s'ils sont dans S_3, la simulation est un succès. Sur un grand nombre de simulations, la fraction de succès convergera vers une valeur $P^*(n_B) \leq P(n_B)$ (l'ensemble S_3 déterminé n'étant pas nécessairement le meilleur). On aura alors une borne inférieure de $P(n_B)$.

On choisit d'étudier un réseau ayant une topologie de grille, où chaque nœud à au plus 4 voisins. Le réseau de la Figure 1 est un exemple de grille. Il faut à présent définir l'ensemble Ctr des zones de contrôle. La zone de contrôle représentée sur la Figure 1 isole un coeur de 3×3 nœuds : on dira donc qu'il s'agit

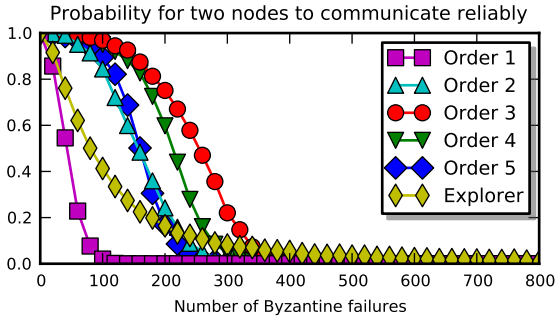

FIGURE 3: Résulats de simulation sur une grille

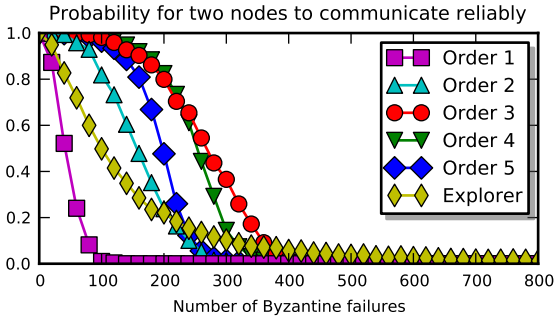

FIGURE 4: Résulats de simulation sur un tore

d'une zone de *largeur* 3. Par la suite, on appelera protocole d'*ordre N* un protocole qui utilise, comme zones de contrôles, toutes les zones possibles de largeur 1 à N. Voir [MT11] pour les définitions formelles de *grille* et d'*ordre*.

La Figure 3 donne les résultats de simulation sur une grille de 100×100 nœuds, pour différents ordres du protocole. On constate qu'un optimum se dessine pour l'ordre 3. La raison est la suivante : plus on augmente l'ordre, plus il y a des zones de contrôle, ce qui augmente les chances de satisfaire les conditions du Théorème 1, mais diminue les chances de satisfaire les conditions du Théorème 2. L'ordre 3 correspond à un équilibre entre ces deux tendances. Des simulations sur un tore (une grille dont les bords sont connectés) donnent des résultats très peu différents (voir Figure 4).

A notre connaissance, aucun autre protocole de tolérance aux fautes Byzantines ne fonctionne sur une topologie aussi faiblement connectée que la grille. La seule exception est le protocole Explorer [NT09], qui utilise des chemins nœud-disjoints pour communiquer entre deux nœuds. Dans le cas d'une grille, on peut facilement modifier ce protocole pour le forcer à utiliser des chemins prédéterminés entre deux nœuds. Ses performances sont représentées sur la Figure 3. Par exemple, si on vise une probabilité $P(n_B) \geq 0.99$, notre protocole permet de tolérer jusqu'à 50 nœuds Byzantins, contre 5 pour Explorer.

4 Conclusion et perspectives

Nous avons proposé une approche adaptée aux réseaux faiblement connecté, et évalué ses performances sur une grille. Cependant, la même méthodologie peut être réutilisée pour n'importe quelle topologie. Par ailleurs, on peut légitimement se demander ce qu'il advient lorsque la taille de la grille tend vers l'infini. Des simulations montrent que les guaranties s'effondrent. Pour pallier à ce problème, nous travaillons actuellement sur une approche qui permet d'étendre les guaranties à une grille potentiellement infinie.

Références

[DFS05] Vadim Drabkin, Roy Friedman, and Marc Segal. Efficient byzantine broadcast in wireless ad-hoc networks. In *DSN*, pages 160–169. IEEE Computer Society, 2005.

[Koo04] Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In Soma Chaudhuri and Shay Kutten, editors, *PODC*, pages 275–282. ACM, 2004.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3) :382–401, 1982.

[MT11] Alexandre Maurer and Sébastien Tixeuil. Limiting byzantine influence in multihop asynchronous networks. *International Conference on Distributed Computing Systems (ICDCS 2012), http ://arxiv.org/abs/1201.5824*, 2011.

[NT09] Mikhail Nesterenko and Sébastien Tixeuil. Discovering network topology in the presence of byzantine nodes. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 20(12) :1777–1789, December 2009.